energies

Article

Learning Latent Representation of Freeway Traffic Situations from Occupancy Grid Pictures Using Variational Autoencoder

Olivér Rákos [1], Tamás Bécsi [1], Szilárd Aradi [1] and Péter Gáspár [2,*]

[1] Department of Control for Transportation and Vehicle Systems, Budapest University of Technology and Economics, 1111 Budapest, Hungary; rakos.oliver@kjk.bme.hu (O.R.); becsi.tamas@kjk.bme.hu (T.B.); aradi.szilard@kjk.bme.hu (S.A.)

[2] Systems and Control Laboratory, Institute for Computer Science and Control, 1111 Budapest, Hungary

* Correspondence: gaspar@sztaki.hu or gaspar.peter@sztaki.mta.hu

Abstract: Several problems can be encountered in the design of autonomous vehicles. Their software is organized into three main layers: perception, planning, and actuation. The planning layer deals with the sort and long-term situation prediction, which are crucial for intelligent vehicles. Whatever method is used to make forecasts, vehicles' dynamic environment must be processed for accurate long-term forecasting. In the present article, a method is proposed to preprocess the dynamic environment in a freeway traffic situation. The method uses the structured data of surrounding vehicles and transforms it to an occupancy grid which a Convolutional Variational Autoencoder (CVAE) processes. The grids (2048 pixels) are compressed to a 64-dimensional latent vector by the encoder and reconstructed by the decoder. The output pixel intensities are interpreted as probabilities of the corresponding field is occupied by a vehicle. This method's benefit is to preprocess the structured data of the dynamic environment and represent it in a lower-dimensional vector that can be used in any further tasks built on it. This representation is not handmade or heuristic but extracted from the database patterns in an unsupervised way.

Keywords: NGSIM; occupancy grid; convolutional variational autoencoder

check for updates

Citation: Rákos, O.; Bécsi, T.; Aradi, S.; Gáspár, P. Learning Latent Representation of Freeway Traffic Situations from Occupancy Grid Pictures Using Variational Autoencoder. *Energies* **2021**, *14*, 5232. https://doi.org/10.3390/en14175232

Academic Editors: Wiseman Yair, Andrea Mariscotti and Rui Xiong

Received: 16 June 2021
Accepted: 13 August 2021
Published: 24 August 2021

1. Introduction

Hierarchical design is the most common approach when designing software for self-driving vehicles. According to this approach, one can think about three main layers: perception, planning, and actuation [1]. Perception combines sensor information to form a model of the world around. In the planning layer, the vehicle model and the environment model are used to plan a route, trajectory, maneuvers, taking into account the specified driving style, destination, and other input instructions. The actuation layer is responsible for the implementation of the planes and gives orders to the actuators. The operation of this and the perception layer are outside the scope of this article. The focus is on the second planning layer, more specifically on behavior prediction. Autonomous vehicles need to make decisions continuously about the maneuvers so that they navigate safely. These decisions can be individual or cooperative based on the traffic system [2].

For the planning layer to function correctly, i.e., create safe, accurate, feasible plans, it must be able to interpret and predict the latent intentions or behaviors of drivers of surrounding vehicles [3]. This task can be captured by trajectory prediction, maneuver detection, and prediction, combining these aspects via multi-modal trajectory prediction. As highway transport is an interactive system, the decisions of vehicle drivers are influenced by all other drivers, and since common road rules must be followed, it can be said to be a strongly coupled system. This means that the intentions of the agents in a traffic situation, or the trajectory itself, can be predicted by using information indicating the inner states of each agent. If these are neglected, one can make limited statements about the future states [4]. In such a strongly coupled system, the past states of a given agent do not encode the internal states of the traffic situation that are required for prediction. This can also be formulated so that we cannot infer the future from the past trajectory of a vehicle without environmental information. Without being exhaustive, behavior prediction is very challenging and attracts great attention in light of these findings. Trajectory patterns can be classified or predicted by Gaussian process regressions [5]. One other famous model is a hierarchical dynamic Bayesian network which is utilized for predicting future patterns about behavior [6].

According to a possible categorization approach, motion prediction systems can be classified into one of the physics-based, maneuver-based, and interaction aware models [7]. The simplest is physics-based, for they consider only the law of physics behind the motion of the vehicles. One gets a more advanced model if it considers the drivers' intentions. These are the maneuver-based models. Interaction-aware models aim to recognize and consider all the inter-dependencies between the vehicles in a traffic situation. In this article, a method is introduced, with the effect of the agents in the traffic situation being considered by an occupancy grid. Such an image has a large dimensionality, so some compression is required. The proven Variational Autoencoder scheme for trajectory copying [8] only could be used for the prediction task if environmental information was included in the input data from which the decoder part can infer suitable predictions about the possible future outcomes. The point of this is to supplement the context vector formed from the trajectory with a "situation" vector that encodes the environmental information. This is how the encoder and decoder are trained to provide the future trajectory at the output. Kim et al. propose a model that predicts the future location of vehicles with 0.5 s, 1 s, and a 2 s time horizon, on an occupancy grid map [9]. In contrast, the goal of this article is only to filter out the most valuable and concise information possible from the location of surrounding vehicles.

The situation vector can be determined in several ways. One approach is structured data about the traffic environment. A novel approach is presented in [10] for predicting the movement of vehicles on the freeway. The output of the model is a multi-modal distribution of trajectories. The input is constructed from the self vehicle (ego) longitudinal and lateral positions appended by the relative positions of six surrounding vehicles: two cars in front of and behind the ego and two more pairs in the adjacent lanes. This input tensor is organized in a specific structured order so that the encoder could learn the proper context representation. Note that the input vectors are embedded using a 64 unit dense layer before the Long-Short Term Memory (LSTM) layer, so the situation encoding representation is 64-dimensional as in this present article. A very similar representation can be found in the research of Feng et al. [11]. They propose a Conditional Variational Autoencoder for intention-based trajectory prediction. The inputs are organized similarly to the previous example. A structured historical observation sequence is taken, containing the EGO's and five surrounding vehicles' displacement, velocity, and longitudinal distance between the ego and the other vehicle. The difference is that the rear vehicle is not included in the structured input since the front vehicle takes no responsibility for the rear according to traffic regularizations.

Agents adjust their path and trajectory by reasoning about their neighbors' movement, and other surrounding agents influence these neighbor agents. Every agent in a freeway may have different neighbors, and, in a crowded traffic situation, the number of correlations will be so high that it cannot be calculated. This is why social pooling is introduced [12] to combine the information from all neighboring states. It was used to handle human trajectory prediction in crowded places. This idea was improved for multi-modal vehicle trajectory prediction in [13] and the authors proposed convolutional social pooling instead of fully connected layers to social-tensors of the LSTM states encoding the historical motion of surrounding vehicles.

Another important way to consider the environment is the occupancy grid. Hoerman et al. introduce a deep convolutional network trained by dynamic occupancy grid maps [14]. The learning-based situation prediction approach utilizes one single neural network. The input is a time series of the data from multiple sensors. The network can segment the static and dynamic areas of the perceivable scene. Cui et al. propose a network architecture that handles the neighbor actors with Convolutional Neural Networks (CNN) [15]. The essence of the method is that they rasterize a birds-eye-view raster image encoding the map of the traffic environment of a given actor. A CNN network processes it, and a raster feature vector is created. The actor's state vector is then concatenated to it and further processed by fully connected layers. Thus, it yields a context vector encoding the actor's state and the surrounding vehicles.

Occupancy grid maps are often used in navigation tasks, without the need for completeness, either for GPS data processing or for the movement of automotive robots or vehicles. Nawaz et al. [16] propose a bidirectional recurrent autoencoder to generate the missing point of a trajectory from GPS data over an occupancy grid map. Small automotive robots need a representation of the environment [17,18] and occupancy grids are suitable for that purpose. Most importantly, occupancy grids are used for long-term prediction of time evolution in a municipal complex traffic system [19]. Deep CNN networks utilize long short-term memories to seize the data's static and dynamic features and focus on the dynamic part for prediction. Park et al. attempt to predict vehicles' future trajectories over an occupancy grid [20]. In the article, an autoencoder architecture analyzes the patterns of past trajectories using LSTM. In the work of Lu et al. [21], a Variational Autoencoder (VAE) neural network is used to encode the front-view visual information of the driving scene and to decode it into a bird-eye view semantic-metric occupancy grid that outperforms the deterministic mapping approach with the flat-plane assumption by more than 12% mean intersection-over-union. These research directions and results suggest that it is worthwhile to extract dynamic information influencing driver decisions from occupancy grids with a VAE neural network.

This paper presents a 2D Convolutional Variation Autoencoder application to copy and compress occupancy grids, thus constructing a situation vector. Deep neural networks received great attention in several fields since they showed promising performance for various tasks in machine learning [22]. In Section 4.1, the preparation is included, and the details of how the dataset of grid images is constructed are described. The methodology and mathematical derivation are described in Section 4.2, and the details of the training in Section 4.3. The discussion of the training results and the quality of the encoder are summarized in Section 5 with a focus on the reconstruction capability on Section 5.1 and the latent space on Section 5.1. Section 5 discusses possible advances in research, new directions, and conclusions.

2. Problem Statement

In this article, we address behavior prediction, an essential part of the analysis and prediction of the behavior of other agents in a traffic situation. It is essential for planning to estimate the possible trajectories of other vehicles and other road users as accurately as possible.

Extracting the environmental information needed for reliable behavior prediction is a very important subtask. Extracting latent information relevant to the prediction from the full information in the occupancy grid using deep learning is an exciting problem. Therefore, a CVAE architecture and a procedure to reduce the occupancy grid dimensionality to 3.125% are presented in this article. The size of the images is 16 times 128 pixels, and the latent space dimension is 64. The value of the pixels covered by a vehicle is 1, while the value of the other pixels is 0. The data preparation is explained in detail in Section 4.1. The model used is explained in Section 4.2, as well as details on training in Section 4.3.

3. Contributions of the Paper

In this paper, a method for the task of depicting a traffic situation is presented. Of the structured data, lidar data, occupancy grid methods, the latter is examined. By compressing and copying the grids, the proposed model learns a representation that can be used as input in other tasks. It can copy an image of 2048 pixels while creating a 64-dimensional situation vector that contains valuable information about the traffic situation at a given time. Four Variational Autoencoder models have been trained for comparison by two different prior distributions and two kinds of training methods. In the variational autoencoder, the decoder distribution is Bernoulli, but the latent space generated by the encoder and the prior distribution give significantly better results in the case of Gaussian than in the case of Bernoulli. Furthermore, more accurate reconstruction and faster convergence can be achieved with Adversarial Training than without it.

4. Solution

This section details the proposed solution to the problem discussed in Section 2, starting with the source of the data and the details of its processing in Section 4.1. The model used and the loss function, and the mathematical considerations related to them are presented in Section 4.2. Finally, details of the training are given in Section 4.3.

4.1. Training Dataset

The data required for the training task were extracted from the NGSIM trajectory database. This database contains vehicle trajectories passing through two U.S. highway sections, the US-101 and I-80 [23,24]. Precise locations, velocities, and acceleration values for vehicles are included every 0.1 s. There are 11.8 million registers, each representing one vehicle in a specific frame of time. The trajectories of I-80 are used for preparation and for creating occupancy grid images. In Figure 1, one can see the I-80 freeway scene from where the data were collected. The occupancy grid images were extracted in the following steps. First, the algorithm goes through each vehicle. All vehicles in the database are considered ego vehicles, and the algorithm iterates through each T time at which ego is found. The resolution of the grid consists of 0.5 m squares. The square in which the extent of the vehicle is included is considered to be occupied. Each square is taken as one pixel, so we get a 1-channel image. In the center of each image is the center of the rear of the ego vehicle. The algorithm then locates any vehicle that is near the ego vehicle at time T and inserts it into the grid. In the lateral (x) direction, a distance of 4 m is taken into account, both to the right and left. In the longitudinal direction (y), a distance of 32 m is taken into account so that the shape of the samples is 16, 128.

4.2. Methodology

Variational Autoencoder and Adversarial Autoencoder neural networks have been successfully trained for the previously defined grid copy and with it for the compression task. In the following, we briefly summarize the theoretical considerations behind this choice. Starting from the probabilistic interpretation of the encoder and decoder, we describe the line of reasoning that led to the choice of the appropriate loss function.

Variational Autoencoders [25] explicitly have a regularization term in the training method because the encoder transforms the input to a distribution over the latent space and not to single point. The decoder receives a sample from that distribution and tries to reconstruct the original data as accurately as possible. This latent distribution could converge to a Dirac-delta without a constraint. In contrast to the regularisation of the VAE, something happens differently in Adversarial Autoencoders (ADVAE). In adversarial training, a discriminator neural network competes with the encoder neural network for conflicting objective [26]. The encoder plays the role of the generation process generating samples from the latent distribution similar to the prior distribution. Meanwhile, the discriminator is trained to find the differences between the generated and prior distribution samples. In other words, the encoder attempts to mislead the discriminator and simultaneously the latter one tries to separate the generated from the priori samples.

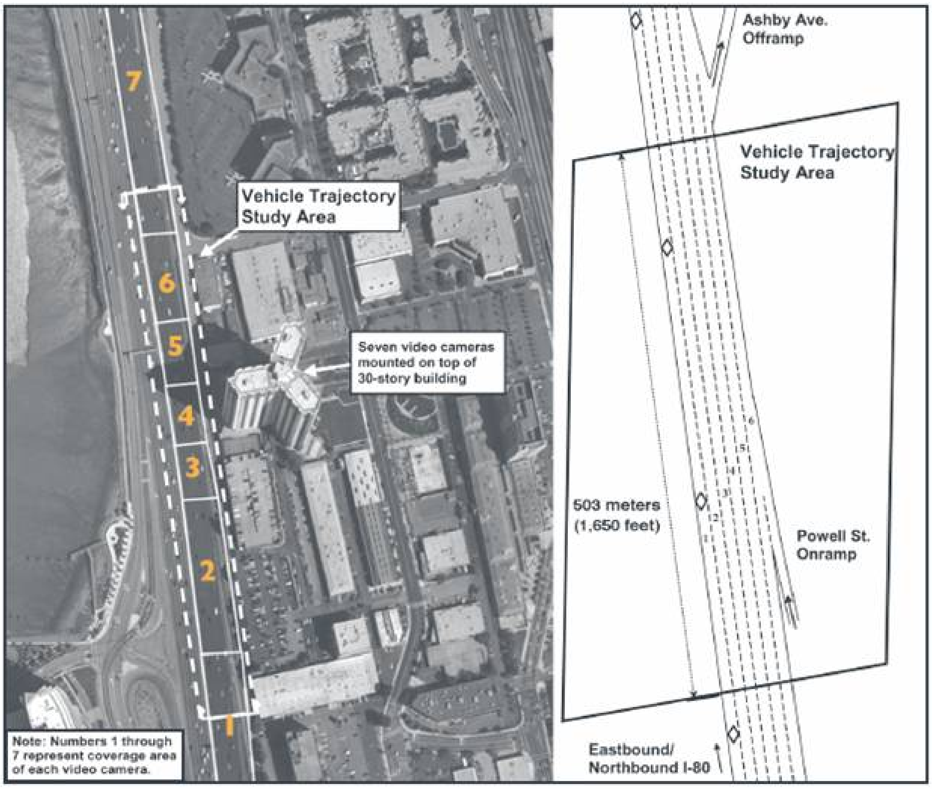

7
6
5
4
3
2
1
Vehicle Trajectory Study Area
Seven video cameras mounted on top of 30-story building
Note: Numbers 1 through 7 represent coverage area of each video camera.
Ashby Ave. Offramp
Vehicle Trajectory Study Area
503 meters (1,650 feet)
Powell St. Onramp
Eastbound/ Northbound I-80

Figure 1. Bird's eye view of Intersection-80 freeway.

By means of variational inference formulation [27], one can define a prior distribution $p(z)$ over the latent space where z is the latent vector. The probabilistic decoder and encoder are defined by conditional probability distributions $p(x|z)$ and $p(z|x)$. The probabilistic decoder denotes the conditional distribution of the decoded variable x given the encoded variable z while the probabilistic encoder is the opposite. Furthermore, one can suppose that the prior is a known distribution which can be easily sampled during training. The sampling process must not prevent the back-propagation of the training loss. Here, two notable distributions are applied as a hypothesis, the Standard Normal and the Bernoulli distributions. The Standard Normal in Equation (1) means that we expect latent vector components to be independent and have zero expected value with unit standard deviation.

$$p(z) = \mathcal{N}(0, \mathbb{I})\,. \tag{1}$$

On the other hand, in Equation (2), our expectation is that the pixels are independent, and have a value of 1 with a probability of 0.5 and 0, respectively:

$$p(z) = \mathcal{B}\left(\frac{1}{2}\mathbb{I}\right). \tag{2}$$

The encoder distribution can be expressed by the Bayesian theorem:

$$p(z|\,x) = \frac{p(x|\,z)p(z)}{p(x)} = \frac{p(x|\,z)p(z)}{\int p(x|\,u)p(u)}\,. \tag{3}$$

The integral is infeasible so one should live with an approximation. This means that $p(z|\,x)$ is approximated by the normal distribution or Bernoulli distribution:

$$q_x^{(N)}(z) \equiv \mathcal{N}(m(x), s(x))\;\; m \in M; s \in S. \tag{4}$$

$$q_x^{(B)}(z) \equiv \mathcal{B}(\theta(x)) \ \theta \in \Theta. \tag{5}$$

The M, S, and Θ function sets are parametrized by encoder neural networks. The optimal approximation is yielded by the optimal (m^*, s^*) function parameters by which the Kullback–Leibler Divergence (KLD) [28] is minimal:

$$(m^*, s^*) = \underset{(m,s) \in M \times S}{\arg\min} \ KL(q_x(z), p(z|\ x)) \tag{6}$$

By the definition of the KLD and Equation (3):

$$(m^*, s^*) = \underset{(m,s) \in M \times S}{\arg\min} \ \mathbb{E}\left(\log(q_x(z)) - \mathbb{E}\left(\log \frac{p(x|\ z) p(z)}{p(x)}\right)\right), \tag{7}$$

where the terms can be factorized further

$$(m^*, s^*) = \underset{(m,s) \in M \times S}{\arg\min} \ \mathbb{E}\Big(\log(q_x(z)) - \mathbb{E}(\log p(z)) - \mathbb{E}(\log p(x|z)) + \mathbb{E}(\log p(x))\Big). \tag{8}$$

The last term is independent of the optimization parameters, and the rest is formulated as

$$(m^*, s^*) = \underset{(m,s) \in M \times S}{\arg\min} \ \mathbb{E}\Big(- \log p(x|\ z) + KL(q_x(z), p(z))\Big). \tag{9}$$

The effect of the second term is the regularization. Minimizing the KLD between the prior and the generated $q_x(z)$ orders the latent space not too far from the origin encourages the encoder to map similar patterns to similar distributions. The first term is the negative log-likelihood of the decoder distribution, which still cannot be computed; therefore, we take advantage of what can be known about the data and what we expect from the generated data. The input image x is a matrix with values of zero and one. This can be considered as every pixel is labeled to one if it is occupied and zero if it is empty. The output of the decoder is between 0 and 1 because of the sigmoid nonlinearity at the output layer, so $f(z)$ can be interpreted as a probability of the original pixel value is 1. Thus, the Bernoulli distribution is a reasonable assumption for the decoder distribution:

$$p(x|\ z) = d(z)^x (1 - d(z))^{1-x}, \ \textit{where } d \in \ D \tag{10}$$

Here, D is a set of functions that can be parametrized by decoder neural networks. The task is to maximize the log-likelihood of the decoder or, in other words, minimize the negative log-likelihood:

$$(m^*, s^*, d^*) = \underset{(m,s,d) \in M \times S \times D}{\arg\min} \ -\mathbb{E}\Big(x \log(f(z)) + (1 - x) \log(1 - f(z))\Big) + KL(q_x(z), p(z)). \tag{11}$$

Thus, the result is the Binary Cross-Entropy (BCE) of the output generated image distribution and the ground-truth image plus the KLD between the latent and prior distribution. Returning to the assumptions about the latent distribution taken in Equations (1) and (2), one can explicitly calculate the second term of Equation (11). Substituting Equation (2) into Equation (11), it yields the loss function for the Variational Autoencoder using Bernoulli distribution as prior:

$$\begin{aligned} L_{Bernoulli} = & \sum_i - \Big(x_i \log(d(z_i)) + (1 - x_i) \log(1 - d_i(z_i))\Big) \\ & + \sum_i \Big(\theta(x_i) \log(\theta(x_i) + (1 - \theta(x_i)) \log(1 - \theta(x_i)) + \log(2)\Big). \end{aligned} \tag{12}$$

Similarly, using Equation (1),

$$
\begin{aligned}
L_{Gauss} = \sum_i -\Big(x_i \log\left(d(z_i)\right) + (1 - x_i)\log\left(1 - d_i(z_i)\right)\Big) \\
+ \frac{1}{2}\sum_i \Big(s(x_i) - m(x_i)^2 - \log(s(x_i)) - 1\Big),
\end{aligned} \tag{13}
$$

one gets the loss function for the Variational Autoencoder with Normal distribution.

The Autoencoders with both prior assumptions also were trained in terms of adversarial training [26]. The reconstruction loss function used for this is the same as the first term in (12) and (13). The regularisation term is ignored. Instead, two other losses come into play: the discriminator term and the generator term. Each can be understood as a binary classification; since the discriminator's output is a single $\delta(z) = p \in [0, 1]$, probability of the input is not generated. The discriminator consists of the batch of latent vectors sampled from the latent distribution generated by the encoder and the same sized batch of vectors from the prior distribution. The expected output is 0 and 1, respectively. The discriminator loss term is formed by the BCE of the expected output and the network's output. It is used to train only the discriminator. Lastly, the generator term formed the same way with the difference that the input is only the batch of the latent vectors, and the discriminators expected output is 1 instead of 0. Then, the BCE loss is back-propagated but used for training the encoder; meanwhile, the discriminator stays intact. This is the concept of competing objectives: the discriminator is trained to separate the latent vectors from priori samples while the encoder is trained to deceive the discriminator, making it mix them up. In equilibrium, the discriminator yields $p = \frac{1}{2}$ for every input.

4.3. Training Details

The schematic architecture of the CVAE is illustrated in Figures 2–4. The input and output layers are the 16 times 128-pixel grids. The encoder for Gaussian prior is in Figure 2. After every transpose convolutional layer in the decoder part and the encoder first segment, a 2D batch normalization layer and Leaky ReLU nonlinearity of parameter 0.2 are applied. The second segment of the encoder applies only one 2D batch normalization layer after the first convolution and Leaky ReLU of parameter 0.2 after the first three. The encoder for the Bernoulli prior is in Figure 3. The differences are that the last layer is a Sigmoid nonlinearity, and there are no two branches of the second segment of complex layers. All parameters are organized in Table 1 and the learning curves are presented on Figure 5.

The decoder reconstructs the initial images from the latent samples, as illustrated in Figure 4. The parameters are in Table 2. Sigmoid nonlinearity is the last layer in the decoder. The discriminator is a Multi Layer Perceptron (MLP) with five complex, fully connected layers. The input dimension is 64, and the hidden layer sizes are 32, 16, and 8. Batch normalization and Leaky ReLU nonlinearity with a parameter of 0.2 are used for better gradient flow and much faster learning. The output layer is one-dimensional and Sigmoid nonlinearity is applied.

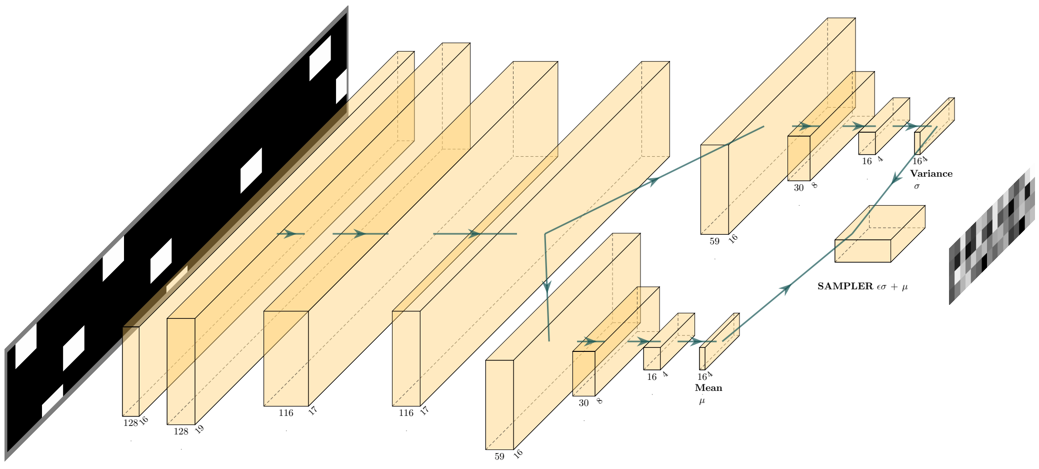


Figure 2. Architecture of the convolutional variational autoencoder's encoder part for Gaussian distribution.

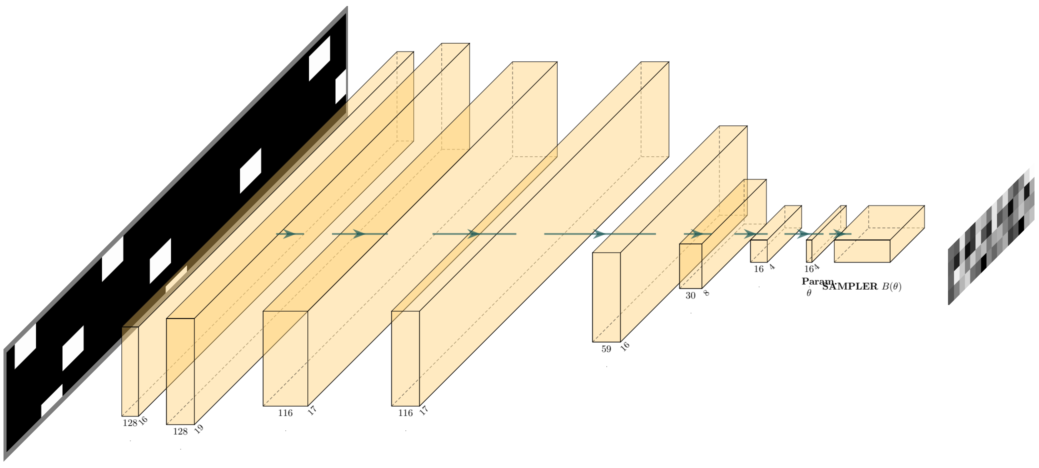


Figure 3. Architecture of the convolutional variational autoencoder's encoder part for Bernoulli distribution.

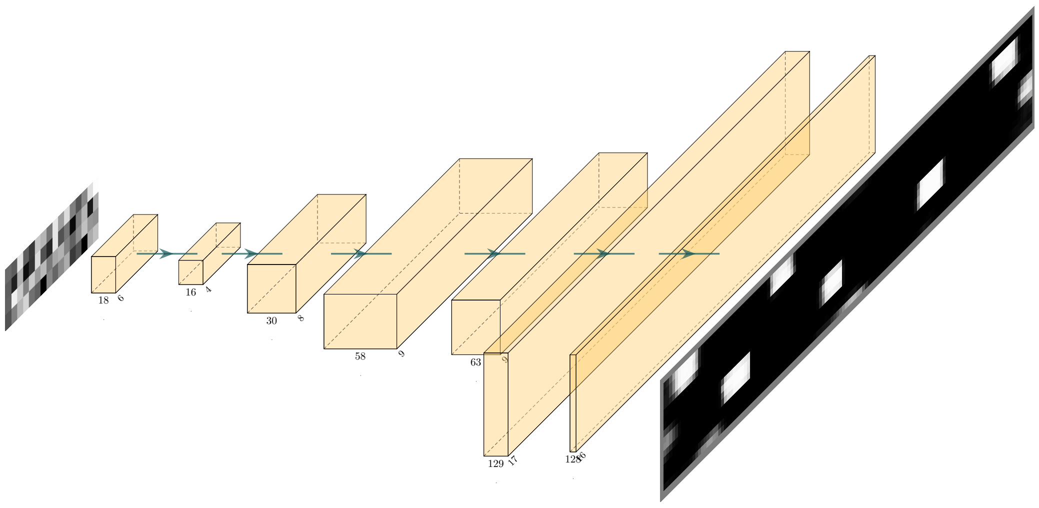


Figure 4. Architecture of the convolutional variational autoencoder's decoder part.

Table 1. Encoder parameters.

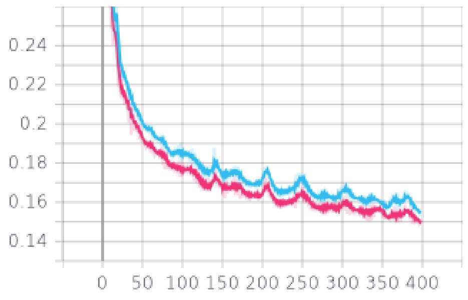
Layers	Channels Input	Output	Kernel	Stride	Padding
First segment					
Conv2D Leaky ReLU (0.2)	1	3	(2, 5)	1	2
Conv2D Batch Norm2D (5) Leaky ReLU (0.2)	3	4	(3, 8)	1	1
Conv2D Batch Norm2D (8) Leaky ReLU (0.2)	5	8	(3, 8)	1	0
Conv2D Batch Norm2D (5) Leaky ReLU (0.2)	8	5	1	1	0
Second segment					
Conv2D Batch Norm2D (5) Leaky ReLU (0.2)	5	5	(4, 4)	(1, 2)	(1, 2)
Conv2D Leaky ReLU (0.2)	5	4	(4, 4)	(2, 2)	(1, 2)
Conv2D Leaky ReLU (0.2)	4	3	(4, 4)	(2, 2)	(1, 2)
Conv2D Sigmoid (With Bernoulli)	3	1	1	1	0

Table 2. Decoder parameters.

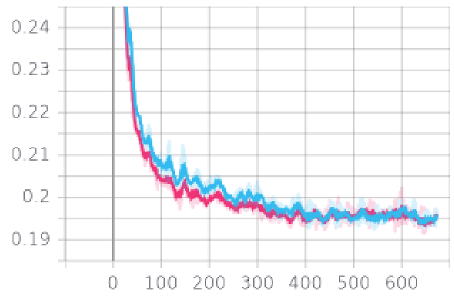
Layers	Channels Input	Output	Kernel	Stride	Padding
ConvTranspose2D Batch Norm2D (4) Leaky ReLU (0.2)	1	4	(3, 3)	1	0
ConvTranspose2D Batch Norm2D (4) Leaky ReLU (0.2)	4	4	(3, 3)	(1, 1)	2
ConvTranspose2D Batch Norm2D (8) Leaky ReLU (0.2)	4	8	(4, 4)	(1, 1)	2
ConvTranspose2D Batch Norm2D (8) Leaky ReLU (0.2)	8	12	(4, 4)	(2, 2)	(1, 2)
ConvTranspose2D Batch Norm2D (8) Leaky ReLU (0.2)	12	8	(3, 8)	(1, 1)	(1, 1)

Table 2. *Cont.*

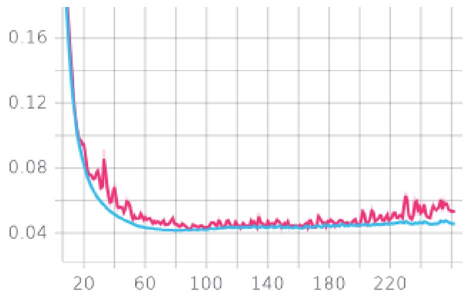
Layers	Channels Input	Channels Output	Kernel	Stride	Padding
ConvTranspose2D Batch Norm2D (4) Leaky ReLU (0.2)	8	4	(3, 8)	(2, 2)	(1, 2) out. pad. = (0, 1)
ConvTranspose2D Sigmoid	4	1	(2, 4)	1	(1, 2)



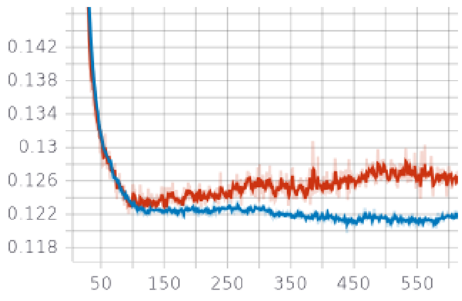
(**a**) VAE with Gaussian distribution



(**b**) VAE with Bernoulli distribution



(**c**) ADVAE with Gaussian distribution



(**d**) ADVAE with Bernoulli distribution

Figure 5. Learning curves. The number of training epochs is on the x-axis. The reconstruction error is on the y-axis. The cyan corresponds to the training reconstruction loss and the pink/red to the validation reconstruction loss.

The training of VAE networks took place in a PyTorch [29] environment, and ADAM optimization [30] is applied with a learning rate of 0.001. The loss function is the sum of BCE and KLD for the reasons discussed in the previous section. Due to better convergence, the KLD term was provided with a multiplication factor of 0.1.

Adversarial training was also performed with ADAM optimization, but the MLP discriminator was trained with Stochastic Gradient Descent (SGD). Separate ADAM optimizer instances are used in the reconstruction step for both the encoder and decoder and for the regularisation step, only the encoder. In all cases, the learning rate was set to be 0.001.

Sampling of the prior and latent distributions yielded by the encoder during training is essential. The sampling method must not disable the backward method calculating the gradient of the losses. Therefore, one cannot sample the $z = \mathcal{N}(\mu, \sigma)$; instead, a random variable is sampled from $\epsilon = \mathcal{N}(0, \mathbb{I})$ standard normal distribution for the Gaussian autoencoders, and it is transformed by the μ mean and λ logarithmic variance values yielded by the encoder

$$z = \sigma\epsilon + \mu = e^{\frac{\lambda}{2}}\epsilon + \mu\,. \tag{14}$$

In the case of Bernoulli, the sampling is not that simple, being a discrete distribution, so the sampling from Bernoulli with θ parameters from the encoder is not a continuous operation, which means there is no gradient to be calculated. A random variable u is sampled from $u \sim \mathcal{U}[0,1]$ uniform distribution and $b \sim \mathcal{B}(\theta)$ is sampled by the rule

$$b \sim \mathcal{B}(\theta) = \begin{cases} 1, & \text{if } u < \theta, \\ 0, & \text{otherwise,} \end{cases} \tag{15}$$

but $b \in \{0,1\}$ does not allow the gradients to be calculated. As a workaround, one should avoid using b by detaching $(b - \theta)$ from the PyTorch variables [29], which becomes just randomized constant C. Adding C to θ gives the same values of b while the gradient calculation is not hindered.

The training of the networks is made using GPU acceleration on hardware "NVIDIA GeForce GTX 1060 6 GB". The training and validation losses are captured in Figure 5. The training of neural networks is inherently slow, but the operation of inference itself is fast since one does not need to calculate gradients and backpropagate the losses through the network. Furthermore, in a current timestep, it does not have a large dataset to process, but a few samples based on the actual traffic situation, so a CPU can deal with it within a reasonable amount of time.

5. Results

In this section, the discussion of the results of the training is summarized. In Section 5.1, the neural network's reconstruction ability is illustrated. The good reconstruction ability shows the good quality of the decoder and that the 64-dimensional code contains information that represents the original picture. The quality of the mapping to latent space is discussed in Section 5.2. It is stated that the slightly different grid samples are mapped to a close-up place. In the time series of grids, similar images are located in the adjacent time steps, but, with interpolation, it is not possible to model temporal dynamics.

5.1. Reconstruction Capability

Figure 6's first column shows 16 grids of $16 * 128$ pixels randomly sampled from the data used for validation. It has different congestion situations and different vehicle sizes. As detailed in previous sections, the rear of the current ego vehicle is located in the grid center, and other vehicles are located in the vicinity.

The rest of the columns are the generated content for these samples, respectively. The copying process is loaded with some noise, which is not surprising for lossy compression. The location of the objects follows the ground truth pattern well. Perfect accuracy is not required anyway since the purpose of the decoder part is merely to learn a proper representation when trained together with the encoder. One significant difference between ADVAE and VAE is that the parameters of the ADVAE encoder are updated twice within an epoch propagating back the discriminator error. Thus, in each epoch, the encoder was encouraged not only to represent the data very well but to generate content similar to the prior distribution. Compared to VAE, an analytically calculated loss function is not used to measure the distance of the distributions but the discriminator network as a function. This may allow for more effective learning. During the training, the model parameters were saved if they produced a lower validation loss than before. Table 3 shows the lowest validation loss values, as well as the loss values haviing taken on the training set with the same model parameters.

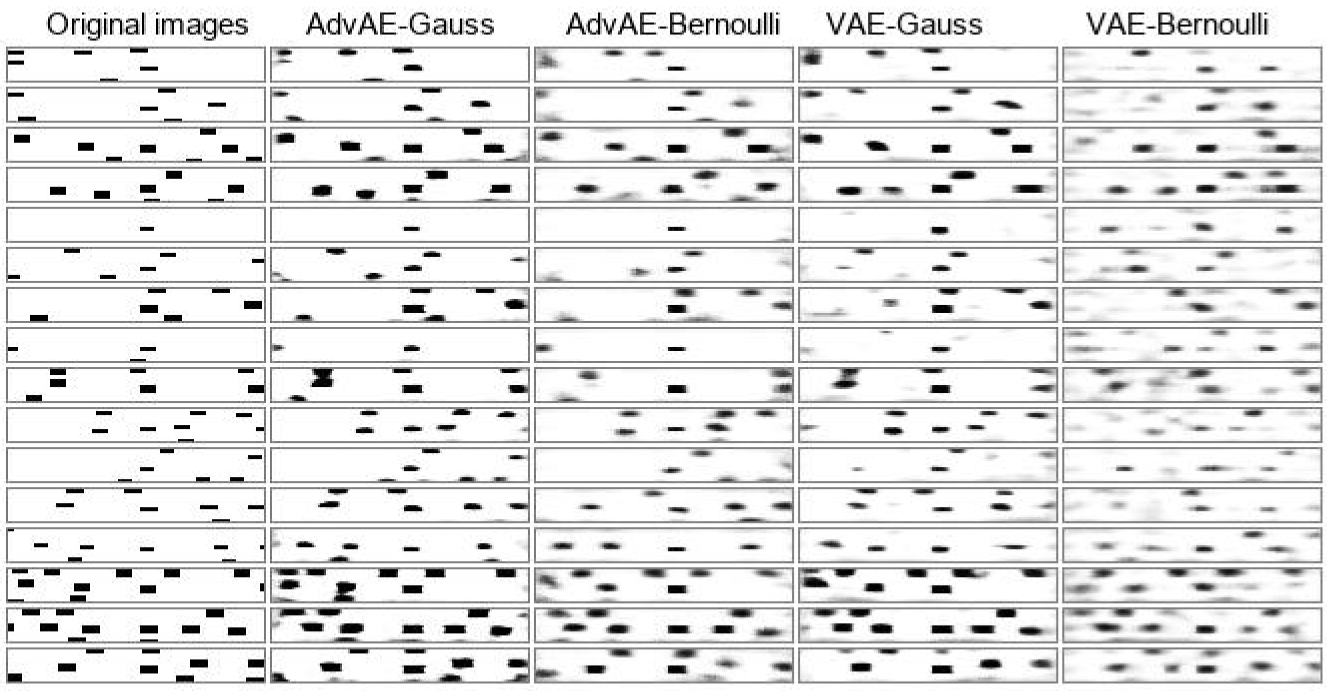


Figure 6. The original occupancy grid samples from the validation dataset.

Table 3. Reconstruction losses on training and validation sets (one below the other) for the models. Two prior distributions (columns) and two methods (rows).

Reconstruciton Loss Train Valid	**Gauss**	**Bernoulli**
ADVAE	0.0406	0.120
	0.0415	0.121
VAE	0.931	0.168
	0.897	0.165

Note that the ADVAE scores are better than the VAE and using Gaussian prior scores better than Bernoulli prior. This supports the hypothesis that the prior distribution is more similar to the Gaussian distribution than the Bernoulli distribution. Although the decoder distribution was intuitively assumed to be Bernoulli, this no longer applies to the latent spatial distribution.

5.2. Latent Space

It is important to note that, during compression and copying, the encoder does not learn any information about the temporal dynamics of the grids. This is not surprising since the training samples are snapshots. What is illustrated in this subsection is that the encoder does not capture temporal dynamics. However, due to the regularizing effect of VAE, similar samples are mapped to similar latent vectors. Consider a continuous time series of an ego vehicle for grid images. Let $G_0 = G(t_0)$ be a grid image at timet_0 and $G_N = G(t_0 + N\Delta t)$ be another one N time steps away. Let their image be $m(G_0)$ and $m(G_N)$. The latent vector m_i is obtained by

$$m_i = \frac{N-i}{N} m(G_0) + \frac{i}{N} m(G_N). \tag{16}$$

Linear interpolation is said to be meaningful if its preimage is the same or similar to the corresponding grid image in the time series. Preimages can be approximated by the decoder $d(m_i)$.

An illustrative example is shown in Figure 7. The second column is the original G_i grids sequence, and the first column is the associated interpolated preimages $d(m_i)$. The first and the last ones from the preimages are equal to the reconstructions of the first and last samples from the original pictures. While we see the real transition of the grids on the right side, on the left, we did not obtain this property by linear interpolation. It is not encoded whether there is an object in a particular location. Grids obtained by interpolation do not have fields for vehicles in an intermediate location. Instead, objects at the beginning and final grid of the sequence gradually disappear and appear, and objects do not continuously transform into each other.

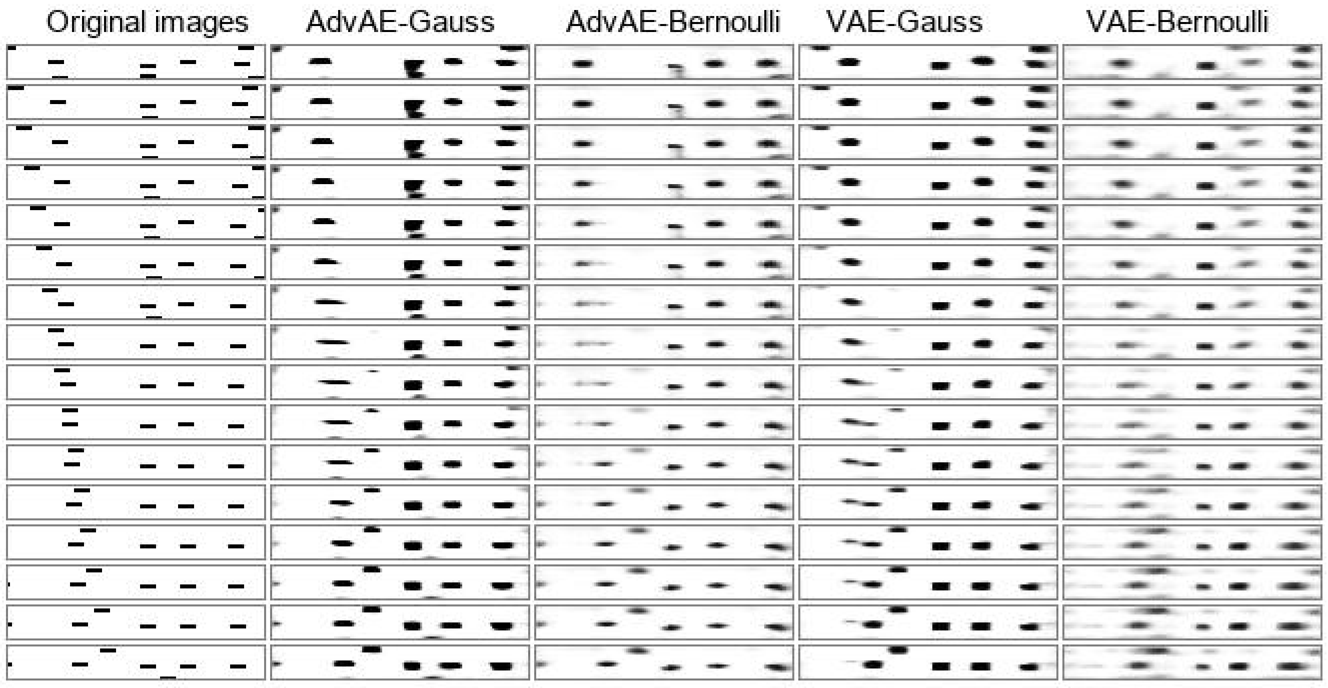


Figure 7. Visualization of interpolation is the latent space. The pictures in the first column show original pictures of time series. The columns from the second to fifth are reconstructed from intermediate latent vectors linearly interpolated between the first and last one.

6. Conclusions and Future Work

The presented CVAE neural network can significantly reduce the dimensionality of occupancy grids. Figures 6 and 7 show the ability of the network to learn the momentary context of the traffic situation but not to interpret it over a time series. The network presented in this article may be helpful for further research in which the traffic situation should also be used in some way, but it is not tied to how these data are plotted. The latent information of these grids is included in the context vector and can thus be used for situation-dependent maneuvers or trajectory predictions.

There are a lot of environmental representation solutions for autonomous vehicles. The current approach is generic and not sensitive for the number of surrounding vehicles, since it takes a fixed window around the ego vehicle, and the other agents are registered within it without a concern for their multiplicity. In our opinion, it can be extended to other topologies as well. The present study provides a proof of concept based on the data of one NGSIM set.

If we also want to encode the dynamic nature of the traffic situation, we need to analyze the time series of occupancy grids. This can also be tested with 3D convolutional Autoencoders. Another method is to train recurrent neural networks, such as LSTM, to encode the time series of latent vectors into a latent space. Comparing two approaches will be useful in developing effective predictive models. The recurrent models can take the series of occupancy grids. A pretrained 2D convolutional encoder would create the compressed form of the grid and pass it to the recurrent unit. To each such hidden code, the positions of the ego vehicle can be concatenated so that the recurrent unit can find a correlation between them. Thus, the recurrent unit is able to encode the trajectory while also processing the environmental information, which is necessary for the prediction.

Author Contributions: Conceptualization, P.G., T.B., and O.R.; methodology, O.R.; software, O.R. and S.A.; validation, S.A. and T.B.,;investigation, P.G.; resources, P.G. and T.B.; data curation, S.A. and O.R.; writing—original draft preparation, O.R. and T.B.; writing—review and editing, S.A. and P.G.; visualization, O.R. and S.A.; supervision, S.A.; funding acquisition, P.G. All authors have read and agreed to the published version of the manuscript.

Funding: The research was supported by the Ministry of Innovation and Technology NRDI Office within the framework of the Autonomous Systems National Laboratory Program.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

CNN	Convolutional Neural Networks
VAE	Variational Autoencoder
CVAE	Convolutional Variational Autoencoder
ADVAE	Adversarial Autoencoder
LSTM	Long-Short Term Memory
ego	self vehicle
KLD	Kullback–Leibler Divergence
BCE	Binary Cross-Entropy

References

1. Geng, X.; Liang, H.; Yu, B.; Zhao, P.; He, L.; Huang, R. A scenario-adaptive driving behavior prediction approach to urban autonomous driving. *Appl. Sci.* **2017**, *7*, 426. [CrossRef]
2. Ploeg, J.; de Haan, R. Cooperative Automated Driving: From Platooning to Maneuvering. In Proceedings of the 5th International Conference on Vehicle Technology and Intelligent Transport Systems-Volume 1: VEHITS, INSTICC, Heraklion, Greece, 3–5 May 2019; SciTePress: Setubal, Portugal, 2019; pp. 5–10. [CrossRef]
3. Llamazares, A.; Molinos, E.J.; Ocaña, M. Detection and Tracking of Moving Obstacles (DATMO): A Review. *Robotica* **2020**, *38*, 761–774. [CrossRef]
4. Rákos, O.; Aradi, S.; Bécsi, T. Lane Change Prediction Using Gaussian Classification, Support Vector Classification and Neural Network Classifiers. *Period. Polytech. Transp. Eng.* **2020**, *48*, 327–333. [CrossRef]
5. Trautman, P.; Krause, A. Unfreezing the robot: Navigation in dense, interacting crowds. In Proceedings of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, Taipei, Taiwan, 18–22 October 2010; pp. 797–803.
6. Dagli, I.; Brost, M.; Breuel, G. Action recognition and prediction for driver assistance systems using dynamic belief networks. In Proceedings of the NODe 2002 Agent-Related Conference on Agent Technologies, Infrastructures, Tools, and Applications for E-Services, Erfurt, Germany, 7–10 October 2002; Springer: Berlin/Heidelberg, Germany, 2002; pp. 179–194.
7. Lefèvre, S.; Vasquez, D.; Laugier, C. A survey on motion prediction and risk assessment for intelligent vehicles. *ROBOMECH J.* **2014**, *1*, 1–14. [CrossRef]
8. Rákos, O.; Aradi, S.; Bécsi, T.; Szalay, Z. Compression of Vehicle Trajectories with a Variational Autoencoder. *Appl. Sci.* **2020**, *10*, 6739. [CrossRef]
9. Kim, B.; Kang, C.M.; Kim, J.; Lee, S.H.; Chung, C.C.; Choi, J.W. Probabilistic vehicle trajectory prediction over occupancy grid map via recurrent neural network. In Proceedings of the 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC), Yokohama, Japan, 16–19 October 2017; pp. 399–404. [CrossRef]
10. Deo, N.; Trivedi, M.M. Multi-Modal Trajectory Prediction of Surrounding Vehicles with Maneuver based LSTMs. In Proceedings of the 2018 IEEE Intelligent Vehicles Symposium (IV), Suzhou, China, 26–30 June 2018; pp. 1179–1184.

11. Feng, X.; Cen, Z.; Hu, J.; Zhang, Y. Vehicle Trajectory Prediction Using Intention-based Conditional Variational Autoencoder. In Proceedings of the 2019 IEEE Intelligent Transportation Systems Conference (ITSC), Auckland, New Zealand, 27–30 October 2019; pp. 3514–3519. [CrossRef]
12. Alahi, A.; Goel, K.; Ramanathan, V.; Robicquet, A.; Fei-Fei, L.; Savarese, S. Social LSTM: Human Trajectory Prediction in Crowded Spaces. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016.
13. Deo, N.; Trivedi, M.M. Convolutional Social Pooling for Vehicle Trajectory Prediction. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, Salt Lake City, UT, USA, 18–23 June 2018.
14. Hoermann, S.; Bach, M.; Dietmayer, K. Dynamic Occupancy Grid Prediction for Urban Autonomous Driving: A Deep Learning Approach with Fully Automatic Labeling. In Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA), Brisbane, QLD, Australia, 21–25 May 2018; pp. 2056–2063. [CrossRef]
15. Cui, H.; Radosavljevic, V.; Chou, F.; Lin, T.; Nguyen, T.; Huang, T.; Schneider, J.; Djuric, N. Multimodal Trajectory Predictions for Autonomous Driving using Deep Convolutional Networks. In Proceedings of the 2019 International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, 20–24 May 2019; pp. 2090–2096.
16. Nawaz, A.; Huang, Z.; Wang, S.; Akbar, A.; AlSalman, H.; Gumaei, A. GPS Trajectory Completion Using End-to-End Bidirectional Convolutional Recurrent Encoder-Decoder Architecture with Attention Mechanism. *Sensors* **2020**, *20*, 5143. [CrossRef] [PubMed]
17. Shi, Y.; Zhang, W.; Yao, Z.; Li, M.; Liang, Z.; Cao, Z.; Zhang, H.; Huang, Q. Design of a Hybrid Indoor Location System Based on Multi-Sensor Fusion for Robot Navigation. *Sensors* **2018**, *18*, 3581. [CrossRef] [PubMed]
18. Gonzalez-Arjona, D.; Sanchez, A.; López-Colino, F.; De Castro, A.; Garrido, J. Simplified Occupancy Grid Indoor Mapping Optimized for Low-Cost Robots. *ISPRS Int. J. Geo-Inf.* **2013**, *2*, 959–977. [CrossRef]
19. Schreiber, M.; Hoermann, S.; Dietmayer, K. Long-Term Occupancy Grid Prediction Using Recurrent Neural Networks. In Proceedings of the 2019 International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, 20–24 May 2019; pp. 9299–9305. [CrossRef]
20. Park, S.H.; Kim, B.; Kang, C.M.; Chung, C.C.; Choi, J.W. Sequence-to-Sequence Prediction of Vehicle Trajectory via LSTM Encoder-Decoder Architecture. In Proceedings of the 2018 IEEE Intelligent Vehicles Symposium (IV), Suzhou, China, 26–30 June 2018; pp. 1672–1678. [CrossRef]
21. Lu, C.; van de Molengraft, M.J.G.; Dubbelman, G. Monocular Semantic Occupancy Grid Mapping With Convolutional Variational Encoder–Decoder Networks. *IEEE Robot. Autom. Lett.* **2019**, *4*, 445–452. [CrossRef]
22. Goodfellow, I.; Bengio, Y.; Courville, A.; Bengio, Y. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016; Volume 1.
23. Colyar, J.; Halkias, J. *Us Highway 80 Dataset*; Federal Highway Administration (FHWA): Washington, DC, USA, 2006.
24. Colyar, J.; Halkias, J. *US Highway 101 Dataset*; Technical Report, FHWA-HRT-07-030; Federal Highway Administration (FHWA): Washington, DC, USA, 2007.
25. Doersch, C. Tutorial on variational autoencoders. *arXiv* **2016**, arXiv:1606.05908.
26. Makhzani, A.; Shlens, J.; Jaitly, N.; Goodfellow, I. Adversarial Autoencoders. arXiv **2015**, arXiv:1511.05644.
27. Blei, D.M.; Kucukelbir, A.; McAuliffe, J.D. Variational Inference: A Review for Statisticians. *J. Am. Stat. Assoc.* **2017**, *112*, 859–877. [CrossRef]
28. Kullback, S.; Leibler, R.A. On information and sufficiency. *Ann. Math. Stat.* **1951**, *22*, 79–86. [CrossRef]
29. Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*; Curran Associates, Inc.: Nice, France, 2019; pp. 8024–8035.
30. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.